Exploratory Data Analysis

Homework 1

Amber Barksdale

Machine Learning
CS-GY 6923
NYU Tandon School of Engineering
February - March 2022

Contents

1 Introduction to Data Set

The data set that was used in this analysis was called “Dry Bean Dataset”, which was found in the University of California Irvine Machine Learning Repository. The data set comes from a research paper titled Multiclass Classification of Dry Beans Using Computer Vision and Machine Learning Techniques (Koklu, M. and Ozkan, I.A., 2020)[2].

1.1 Why dry beans?

Dry beans are a strong commodity in Turkey, where the research was conducted. The quality of dry beans are obviously important, however, it would require an incredible amount of labor if every bean had to be manually inspected. Thus, there is a great need for an accurate and efficient automatic system that can determine the type of bean and its quality, in order to keep up with demand and keep costs down. There are several types of dry beans in Turkey, but only seven of them were used in this research. Factors that were taken into account were: form, shape, type, and structure. All of these factors play into how profitable the bean will be in the market. The authors of the paper developed a computer vision system to distinguish the seven types of beans - all of which have very similar features - in attempts to obtain uniform seed classification.

1.2 Basic data dimensions

There are 13611 instances in the data set and a total of 17 attributes the authors considered. The attributes are seen in Table 1. The seven classes are listed in Table 2.

2 Checking Balance

Checking for balanced data is important as imbalanced data may cause the model in the future to fail. There is a difference between the distribution itself being imbalanced versus the data not capturing the true distribution (as an example, having many more instances of one particular class - but not because there are actually more of that class)[1]. One way to estimate imbalance is just by seeing the number of instances per class.

```
table(drybeans$Class)

BARBUNYA   BOMBAY     CALI  DERMASON    HOROZ    SEKER     SIRA
    1322      522     1630      3546     1928     2027     2636
```

We can see that the bean type “Bombay” has far fewer instances than the rest. A more thorough check can help determine whether the classes are indeed imbalanced. In the following code block, we check whether the ratio of the number of instances in a class is less than 1:3. If that is the case, then there is an imbalance in the data.

ATTRIBUTE	DESCRIPTION
Area (A)	The area of a bean zone and the number of pixels within its boundaries.
Perimeter (P)	Bean circumference is defined as the length of its border.
Major axis length (L)	The distance between the ends of the longest line that can be drawn from a bean.
Minor axis length (l)	The longest line that can be drawn from the bean while standing perpendicular to the main axis.
Aspect ratio (K)	Defines the relationship between L and l.
Eccentricity (Ec)	Eccentricity of the ellipse having the same moments as the region.
Convex area (C)	Number of pixels in the smallest convex polygon that can contain the area of a bean seed.
Equivalent diameter (Ed)	The diameter of a circle having the same area as a bean seed area.
Extent (Ex)	The ratio of the pixels in the bounding box to the bean area.
Solidity (S)	Also known as convexity. The ratio of the pixels in the convex shell to those found in beans.
Roundness (R)	Calculated with the following formula: $4\pi A/P^2$
Compactness (CO)	Measures the roundness of an object: Ed/L
ShapeFactor 1 (SF1)	
ShapeFactor 2 (SF2)	
ShapeFactor 3 (SF3)	
ShapeFactor 4 (SF4)	
Class	

Table 1: The 17 attributes used to define the beans.

Seker	Barbunya
Bombay	Cali
Dermosan	Horoz
Sira	

Table 2: The seven types of beans used in the study.

```
TBL <- table(data$Class)

if(any(TBL<(nrow(data)*1/7*1/3) | any(TBL>(nrow(data)*1/7*3)))) {
    imbalanced_classes=names(TBL)
    [TBL<(nrow(data)*1/7*1/3) | TBL>(nrow(data)*1/7*3)]
    print("Imbalanced␣Classes:")
    print(imbalanced_classes)
}
[1] "Imbalanced␣Classes:"
[1] "BOMBAY"
```

There is one imbalanced class - the Bombay type of bean. This means we will have to handle the data in a different way in the future. However, if we notice that the Bombay bean is starkly different from the other types of beans, then we may be able to keep the data in.

2.1 Handling imbalanced data

There are a handful of methods we can use to handle imbalanced data in the future if we wish to build a model based on this data set.

2.1.1 Oversampling

The oversampling technique increases the number of samples from a "rare" class. An advantage of this is that no information from the original data is lost. However, this may lead to over-fitting[4].

2.1.2 Downsampling

An opposite technique aims to decrease the number of samples in a class that is large. This results in losing potentially useful information from the original dataset[4].

2.1.3 Other options

Other techniques to balance data include: feature selection, Cost-Sensitive Learning (CSL), ensemble learning techniques, and using the right evaluation metrics[4].

If the next step were to prepare this data set for building a model, then one of these methods would be chosen to account for the fact that there are far fewer samples of Bombay than any other bean type.

3 Descriptive Statistics

In this section, we will run some simple commands in order to get basic descriptive statistics about the data set.

3.1 Missing values

First, we will check whether there are any missing values in the data.

```
empty <- sapply(drybeans, function(x) all(is.na(x) | x==''))

Area      Perimeter  MajorAxisLength  MinorAxisLength     AspectRation
FALSE     FALSE      FALSE            FALSE               FALSE
Eccentricity    ConvexArea    EquivDiameter   Extent        Solidity
FALSE           FALSE         FALSE           FALSE         FALSE
roundness   Compactness     SF1    SF2     SF3     SF4     Class
FALSE       FALSE           FALSE  FALSE   FALSE   FALSE   FALSE
```

Luckily, there are no missing values. This means we will not need to decide whether to remove or impute any values, and can move on to statistics and visualizations.

3.2 Visualization

With the descriptive statistics and visualizations, we can start to notice common values, unusual values, possible subgroups, patterns, and anything else that might be of interest. As the main methods of determining the difference between the beans is a measurement of their physical shape, we will determine the average Perimeter, Major Axis Length, and Minor Axis Length for each class.

```
library(dplyr)
newdata <- group_by(drybeans, Class)
newdata <- summarize(newdata, mean_perim=mean(Perimeter, na.rm=TRUE))
newdata

PERIMETER AVERAGES:
# A tibble: 7    2
  Class      mean_perim
  <chr>           <dbl>
1 BARBUNYA        1046.
2 BOMBAY          1586.
3 CALI            1058.
4 DERMASON         665.
5 HOROZ            920.
6 SEKER            728.
7 SIRA             796.
```

Here we can see the beans are separated into two general groups: small and large. However, notice there are some bean perimeter averages that could be considered both small and large. Perimeter, therefore, is a useful attribute but potentially is not enough to be able to accurately classify the bean. In order to visualize these averages better, we will make a boxplot to show the mean, quartiles, and any outliers. We will use the library **ggplot2** to make most of the visualizations.

```r
library(ggplot2)
f <- ggplot(drybeans, aes(Class, Perimeter)) + geom_boxplot()
f
```

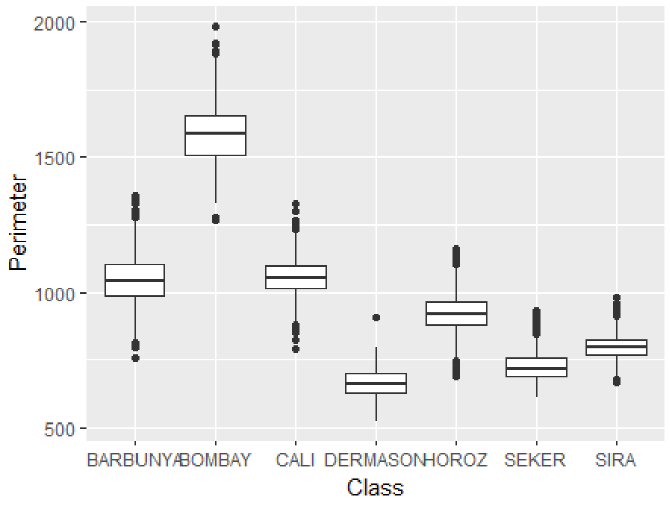

Figure 1: The mean, first and third quartiles, and any min/max outliers for the perimeter for each class. The seed type Bombay has a considerably larger perimeter than all the other bean types.

The box plots show that the Bombay type of bean seems considerably larger than the other types. We also notice that "roundness" may not be an attribute that we can reliably use as there is a wide range among many different types of beans. Plotting attributes against each other can also help visualize any relationships between them.

Notice that the Bombay bean has been shown to be quite different from all the other beans in a variety ways. With such a strong difference, it would be relatively easy to classify this bean. The visualizations have reinforced that the data is imbalanced and in order to prepare the dataset for the future, this would have to be addressed.

4 Collinearity

4.1 Variance Inflation Factor (VIF)

There is possibility of multicollinearity, especially after visualizing many of the variables against each other. If the degree of correlation is high enough, then this may cause problems in the model. However, VIF (Variance Inflation Factor), while a useful tool, does not work on classification - in order to do so, the data must be squeezed into a linear regression model. Due to a number of issues, running VIF was unsuccessful.

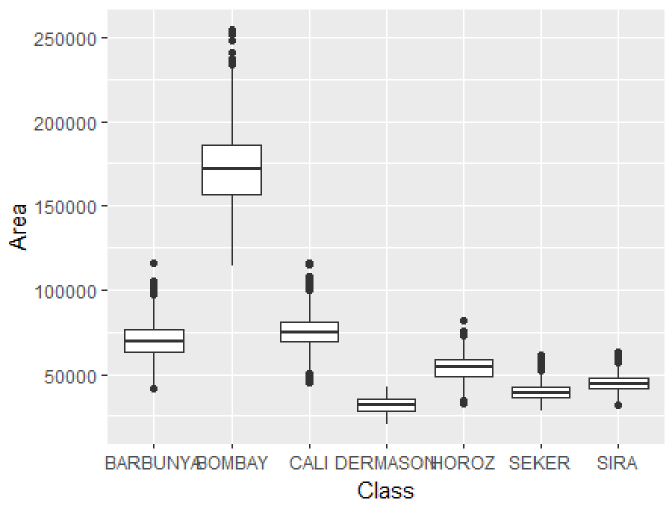

Figure 2: The mean, first and third quartiles, and any min/max outliers for the area of each class. Again, Bombay seems to be much larger than the other beans, and has the widest variety in area.

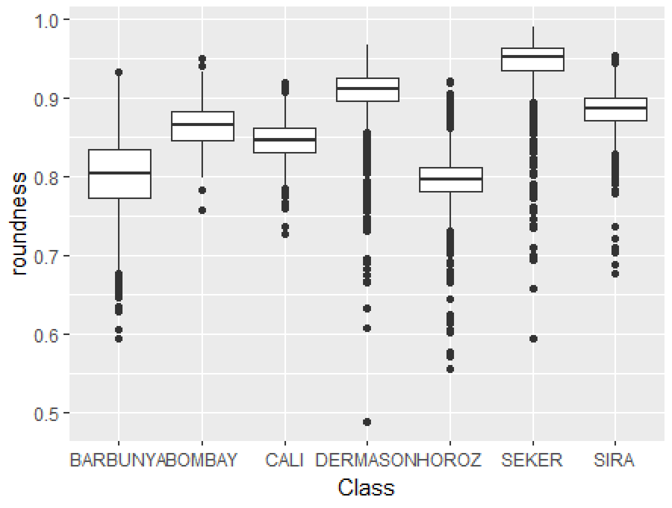

Figure 3: Roundness, which was calculated by $4\pi A/P^2$, for each class. The bean types Dermason, Horoz, and Seker seem to have a wide variety of roundness.

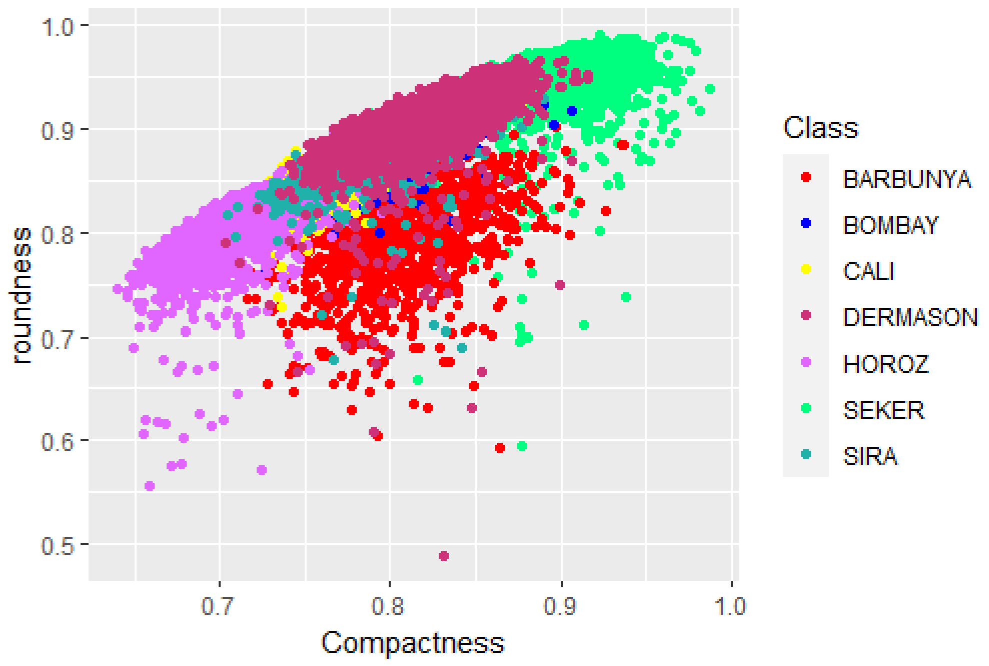

Figure 4: Compactness vs. Roundness for each class. We note there is a general trend, so these variables seem to have a strong relationship. Although it is important to notice there are a handful of outliers. These outliers may need to be removed before training or running a model.

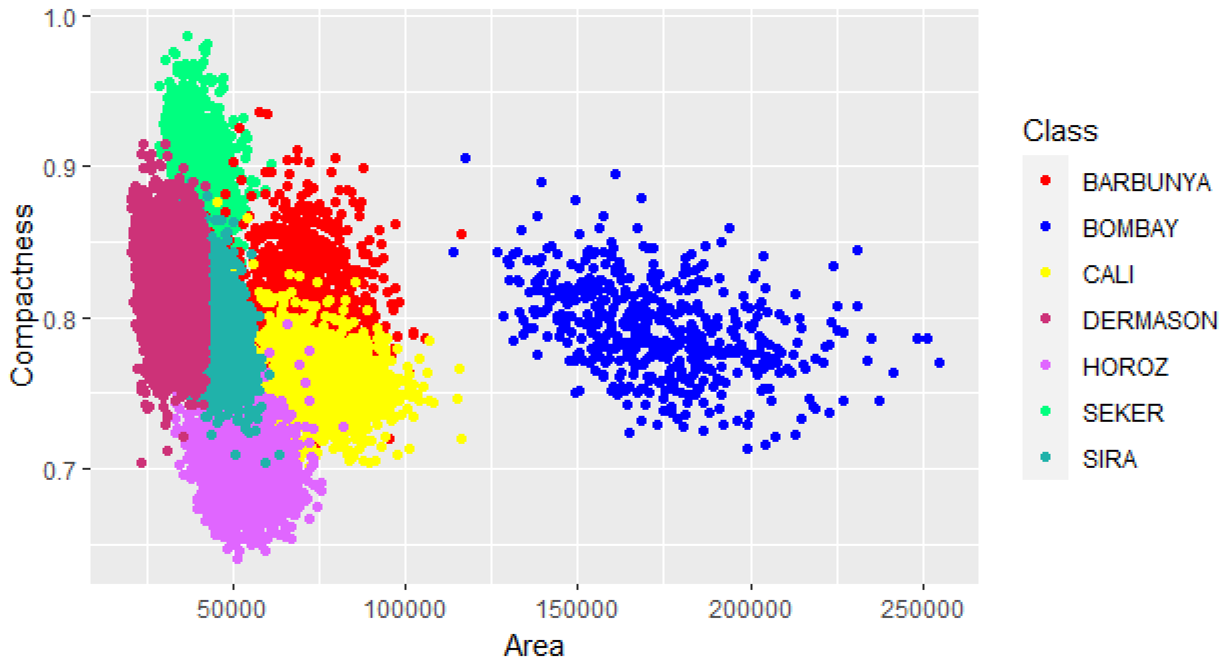

Figure 5: Area vs. Compactness. There seem to be two distinct groups, and Bombay makes up the majority, if not all, of the samples in the right group.

4.2 Is checking multicollinearity necessary?

However, there is argument[3] that checking for multicollinearity is not as important in logistic regression. If the goal is to understand the impact independent variables have on the dependent variable(s), then the presence of strong multicollinearity will make this task impossible. If the goal is to improve the accuracy of prediction, then there is less importance on how the variables have impact on one another - as long as the prediction is correct. In general, multicollinearity does not affect the predictive power of the model, but may affect the coefficients; in the majority of machine learning applications, the coefficients are not the priority, rather the loss of the model prediction is. Therefore, being unable to perform VIF on the dry bean dataset is less of a concern. What is important, though, is to perform standardization and / or normalization.

5 Outliers

It is also important to check for outliers and determine whether those data points should be kept, removed, or imputed. We will check for any data points that lie outside three standard deviations of the mean. Looping through each numerical column, we calculate the mean and standard deviation (σ). Then we calculate the upper and lower cutoffs, which are the mean of the specific column $\pm 3\sigma$. Then we check for any data points outside that range and append it to the outliers list. After looping through each numerical column, we count only the unique values. The number of unique outliers was 1124. Whether or not we wish to keep the outliers will depend on the importance of specific factors, which we will determine after performing Principal Component Analysis (PCA). It may be necessary to keep outliers in features that prove to be important in predicting the bean type.

```r
# Initialize a variable
outliers = c()
for(i in 1:16){
    data_mean = mean(db_numeric[,i])
    data_sd = sd(db_numeric[,i])
    low = data_mean - 3*data_sd
    up = data_mean + 3*data_sd
    index = which(db_numeric[,i] < low | db_numeric[,i] > up)
    outliers = c(outliers, index)
}

# Remove duplicates
outliers = unique(outliers)

# 1124 outliers
```

6 Standardization and Normalization

6.1 Min-Max vs. Z-Score

Normalizing data is necessary prior to dimensionality reduction techniques such as Principal Component Analysis (PCA). We have a few options, two of them being Min-Max Normalization and Z-Score Normalization. With Min-Max Normalization, we are guaranteed that all features will have the exact same scale. One drawback is that outliers are not handled well. Z-Score Normalization handles outliers but not all the data will have the exact same scale. We wish to perform PCA after standardizing the data, and the algorithm will give higher importance to higher numerical values, so we want all the features at the same scale. Thus, we will use Min-Max Normalization[5] on this dataset prior to Principal Component Analysis.

```r
# Define the normalization function
min_max_norm <- function(x) { (x - min(x)) / (max(x) - min(x)) }

# Only take into consideration the numerical columns
db_norm <- as.data.frame(lapply(drybeans[1:16], min_max_norm))

# Add the class column back in
db_norm$Class <- Dry_Bean_Dataset$Class
```

In Z-Score Normalization, the means all become 0 - but in Min-Max Normalization, this is not the case, as each numerical feature had a different scale and the goal was to normalize relative to each other, not to the zero mean.

7 Dimensionality Reduction

Now that the data has been normalized, we can perform dimensionality reduction. Dimensionality reduction is very useful, especially in situations where there are several features. This data set has 17 attributes, some of which we have seen hold more weight than others.

7.1 PCA Rotation

We will also look at the PCA rotation to see which features matter the most when it comes to determining which bean is which. In the rotation, we took the absolute value, then sorted, and printed out the names in order to see the feature that had the most importance. While a negative value suggests an inverse correlation, the magnitude is what suggests the strength of the relationship, which is why the absolute value was used.

```r
# Determine the columns that are numeric (all except Class)
numeric_vars <- c(
  "Area", "Perimeter", "MajorAxisLength", "MinorAxisLength",
  "AspectRation", "Eccentricity", "ConvexArea", "EquivDiameter",
  "Extent", "Solidity", "roundness", "Compactness",
  "ShapeFactor1", "ShapeFactor2", "ShapeFactor3", "ShapeFactor4"
)

# Slice the data so it only contains the numeric columns
db_numeric <- db_norm[, numeric_vars]

# Perform PCA on the numeric data
res_pca <- prcomp(db_numeric, scale = TRUE)

# Print out (sorted, abs val) the features by name
res_pca$rotation[,1]
```

The results of the PCA were as follows:

```
Area 0.28245796              Perimeter 0.31089112
MajorAxisLength 0.32582398   MinorAxisLength 0.23619938
AspectRation 0.22929833      Eccentricity 0.23152605
ConvexArea 0.28319989        EquivDiameter 0.29748384
Extent -0.05980796           Solidity -0.14301631
roundness -0.24816481        Compactness -0.23837800
ShapeFactor1 -0.22131890     ShapeFactor2 -0.31462459
ShapeFactor3 -0.23898330     ShapeFactor4 -0.19800943
```

The results of the rotation were as follows:

```
names(sort(abs(res_pca$rotation[,1]), decreasing=TRUE))

1. "MajorAxisLength" 2. "ShapeFactor2" 3. "Perimeter"
4. "EquivDiameter" 5. "ConvexArea" 6. "Area"
7. "roundness" 8. "ShapeFactor3" 9. "Compactness"
10. "MinorAxisLength" 11. "Eccentricity" 12. "AspectRation"
13. "ShapeFactor1" 14. "ShapeFactor4" 15. "Solidity" 16. "Extent"
```

After seeing the results of PCA and the rotation, we can consider which dimensions should be utilized. The top three factors are Major Axis Length, Shape Factor 3, and Perimeter. When building a model, these three features would likely be the strongest indicator of what type of bean is being classified. It is interesting that one of the "Shape Factors" turned out to be a major contributor to classifying bean type. This is a feature that was specifically created and calculated by the authors of the study, so without further research on what exactly ShapeFactor3 represents, it may be difficult to extrapolate to other bean-classifying data.

8 Conclusion

This data set seems to be a viable option for a classification model. This is unsurprising as it was indeed used to build several different models - in the paper from which the data originated. There was a great enough pressure to develop successful classification models as the quality of dried beans are a cultural and economic staple in Turkey, where the research took place. Certainly some more data cleaning and modification would have to take place prior to building a model; the research paper itself would also serve as a useful tool to determine what needs to be done.

9 R Script

```r

library(readxl)
library(ggplot2)
library(dplyr)

print("Reading in the Excel file...")
drybeans <- read_excel("Dry_Bean_Dataset.xlsx")

table(drybeans$Class)

TBL <- table(drybeans$Class)

print("Checking for imbalanced data...")
# CHECK FOR IMBALANCED DATA
if(any(TBL<(nrow(drybeans)*1/7*1/3) |
any(TBL>(nrow(drybeans)*1/7*3)))) {
  imbalanced_classes=names(TBL)[TBL<(nrow(drybeans)*1/7*1/3) |
  TBL>(nrow(drybeans)*1/7*3)]
  print("Imbalanced Classes:")
  print(imbalanced_classes)
}

print("Basic descriptive statistics...")
newdata <- group_by(drybeans, Class)
newdata <- summarize(newdata, mean_perim =
mean(MajorAxisLength, na.rm=TRUE))
newdata
newdata2 <- group_by(drybeans, Class)
newdata2 <- summarize(newdata2, mean_perim =
mean(MinorAxisLength, na.rm=TRUE))
newdata2

print("Visualizing data...")# SOME VISUALIZATION (nonexhaustive)
```

```

ggplot(data = drybeans, aes(MajorAxisLength, AspectRation,
color = Class)) + geom_point() + scale_color_manual(values =
c("BARBUNYA" = "red", "BOMBAY" = "blue", "CALI" = "yellow",
"DERMASON"= "violetred3", "HOROZ" = "mediumorchid1",
"SEKER" = "springgreen1", "SIRA" = "lightseagreen"))


f <- ggplot(drybeans, aes(Class, Perimeter)) + geom_boxplot()
f
f <- ggplot(drybeans, aes(Class, MajorAxisLength)) + geom_boxplot()
f
f <- ggplot(drybeans, aes(Class, MinorAxisLength)) + geom_boxplot()
f
f <- ggplot(drybeans, aes(Class, roundness)) + geom_boxplot()
f
f <- ggplot(drybeans, aes(Class, Compactness)) + geom_boxplot()
f
f <- ggplot(drybeans, aes(Class, Area)) + geom_boxplot()
f

print("Check␣for␣missing␣values...")
# CHECK FOR MISSING VALUES

allmisscols <- sapply(drybeans, function(x) all(is.na(x) | x ==''))
allmisscols

print("Min-max␣normalization...")
# MIN-MAX NORMALIZATION
min_max_norm <- function(x) { (x - min(x)) / (max(x) - min(x)) }

db_norm <- as.data.frame(lapply(drybeans[1:16], min_max_norm))
head(db_norm)
db_norm$Class <- drybeans$Class
head(db_norm)

print("Get␣only␣numeric␣data␣for␣PCA...")
# GET ONLY NUMERIC DATA
numeric_vars <- c(
  "Area", "Perimeter", "MajorAxisLength", "MinorAxisLength",
  "AspectRation", "Eccentricity", "ConvexArea", "EquivDiameter",
  "Extent", "Solidity", "roundness", "Compactness",
  "ShapeFactor1", "ShapeFactor2", "ShapeFactor3", "ShapeFactor4"
)
db_numeric <- db_norm[, numeric_vars]
ncol(db_numeric)

```

```
print("Perform␣PCA...")
# PCA
res_pca <- prcomp(db_numeric, scale = TRUE)

print("See␣PCA␣rotation...")
# PCA ROTATION
res_pca$rotation[,1]
names(sort(abs(res_pca$rotation[,1]), decreasing=TRUE))

print("Check␣for␣outliers...")
# OUTLIERS
# Initialize a variable
outliers = c()
for(i in 1:16){
  data_mean = mean(db_numeric[,i])
  data_sd = sd(db_numeric[,i])
  low = data_mean - 3*data_sd
  up = data_mean + 3*data_sd
  index = which(db_numeric[,i] < low | db_numeric[,i] > up)
  outliers = c(outliers, index)
}

# Remove duplicates
outliers = unique(outliers)
print(paste("Number␣of␣outliers:",length(outliers)))
```

10 References

References

[1] Valentin Calomme. *Why do we need to handle data imbalance?* URL: https://datascience.stackexchange.com/questions/24392/why-do-we-need-to-handle-data-imbalance.

[2] Murat Koklu and Ilker Ali Ozkan. "Multi-class classification of dry beans using computer vision and machine learning techniques". In: *Computers and Electronics in Agriculture* 174 (2020), p. 105507. ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2020.105507. URL: https://www.sciencedirect.com/science/article/pii/S0168169919311573.

[3] Sycorax. *Why is multicollinearity not checked in modern statistics/machine learning.* URL: https://stats.stackexchange.com/questions/168622/why-is-multicollinearity-not-checked-in-modern-statistics-machine-learning/168631#168631.

[4] Himanshu Tripathi. *What Is Balanced And Imbalanced Dataset?* URL: https://medium.com/analytics-vidhya/what-is-balance-and-imbalance-dataset-89e8d7f46bc5.

[5] Zach. *How to Normalize Data in R.* URL: https://www.statology.org/how-to-normalize-data-in-r/.